Computer Conservation Society

Aims and objectives

The Computer Conservation Society (CCS) is a co-operative venture between the British Computer Society and the Science Museum of London.

The CCS was constituted in September 1989 as a Specialist Group of the British Computer Society (BCS). It thus is covered by the Royal Charter and charitable status of the BCS.

The aims of the CCS are to

- ◇ Promote the conservation of historic computers and to identify existing computers which may need to be archived in the future
- ◇ Develop awareness of the importance of historic computers
- ◇ Encourage research on historic computers and their impact on society

Membership is open to anyone interested in computer conservation and the history of computing.

The CCS is funded and supported by a grant from the BCS, fees from corporate membership, donations, and by the free use of Science Museum facilities. Membership is free but some charges may be made for publications and attendance at seminars and conferences.

There are a number of active Working Parties on specific computer restorations and early computer technologies and software. Younger people are especially encouraged to take part in order to achieve skills transfer.

The corporate members who are supporting the Society are Bull HN Information Systems, Digital Equipment, ICL, Unisys and Vaughan Systems.

Resurrection

The Bulletin of the Computer Conservation Society

ISSN 0958 - 7403

Number 15

Summer 1996

Contents

News Round-Up

Tony Sale has completed his construction of the replica Colossus at Bletchley Park. It was inaugurated by HRH Duke of Kent at a special ceremony on 6 June, which was appropriately the 52nd anniversary of D-Day. Interested readers can go and see it in operation at weekends — see Forthcoming Events for details.

- 101010101 -

A detailed account of this project, written by the editor of *Resurrection*, appeared in the 6 June issue of *Computer Weekly*.

- 101010101 -

Konrad Zuse, one of the small band of people involved in developing computer systems using electromechanical technology in the late 1930s and early 1940s, died just before Christmas at the age of 85.

- 101010101 -

Readers who enjoyed Donald Kershaw's article "Experiences with Pegasus 1" in issue 14 will be sorry to learn that Stephen Vajda, leader of the Mathematics Group at the Admiralty Research Laboratory when Dr Kershaw moved there in 1957, died on 10 December 1995 aged 94.

- 101010101 -

The Leo Reunion Society is planning to change its name to the Leo Computers Society, and to publish a constitution. The next reunion is scheduled for April 1997. More details from Peter Byford on 01920 463804.

- 101010101 -

Members are reminded that the Society now has an electronic archive, from where members with access to the Internet and who can use FTP (File Transfer Protocol) may download files, including the current and all past issues of *Resurrection*. To access the archive, connect as an anonymous user to **ftp.cs.man/ac/uk** and change to directory **pub/CCS-Archive**...

- 101010101 -

... and more for those with modems: Jon Agar of the National Archive for the History of Computing, Manchester, has written to *Resurrection* to let members know that an e-mail distribution list has been set up to carry information about news and events relating to the history of computing in the UK. He says the list, **history-of-computing-uk**, is of particular interest to historians of computing, business historians and museum curators. Readers wishing to join should send the message *join history-of-computing-uk (your first name) (your second name)* to **mailbase@mailbase.ac.uk**. For example: *join history-of-computing-uk Charles Babbage.*

- 101010101 -

The Elliott 401 Working Party has been advised that the new computer room with strengthened floor is complete at Blythe House, apart from the carpet tiles. When all is ready, the Working Party will meet to supervise the transfer of the machine into its new area, and assemble the cabinets onto the plinths.

e-mail addresses

Readers of *Resurrection* who wish to contact committee members via electronic mail may do so using the following addresses.
Chris Burton: chris@envex.demon.co.uk
Martin Campbell-Kelly: mck@dcs.warwick.ac.uk
Dan Hayton: Daniel@newcomen.demon.co.uk
Adrian Johnstone: adrian@dcs.rhbnc.ac.uk
Tony Sale: t.sale@qufaro.demon.co.uk
Robin Shirley: r.shirley@surrey.ac.uk
Doron Swade: d.swade@ic.ac.uk

Society Activity

Chris Burton

London Pegasus Working Party

Two all day meetings and a few short *ad hoc* meetings have taken place. We were able to restore the machine to working order again after tracking down a faulty package in the drum clock circuitry. The machine then ran the Initial Orders and some application programs, but not the Engineer's Tests. The problem was traced to one of the store locations which had a 336-digit delay line instead of the correct 42-digit line! By analysing the past movement of packages in and out of the machine, it became apparent that I made this swap nearly a year ago (by mistake!), and the error has only just come to light. We have now started to weed out marginal packages to bring the machine back to a healthy state prior to its move to a new area during the next year. We also discovered at our last meeting that we seem to have an inadequate incoming power feed somewhere, as the main supply cable and fuses are getting very hot.

Small-Scale Experimental Machine

The North-West Pegasus Working Party is now working as the Technical Team on the SSEM Rebuild Project. The project was launched officially on 5 March at the Town Hall, Manchester. Here the University, the City and the Museum of Science and Industry all declared their support for our efforts, and ICL High Performance Systems, West Gorton, announced they would be sole sponsors. One rack of equipment was shown working with two lines of 32 bits stored on a cathode ray tube. Three pioneers — Tom Kilburn, Geoff Tootill and Alec Robinson — were present.

We have been given space to build the replica in the Manchester Computing computer hall in the University, and several empty racks are in place. The power supply system is coming together in the ICL workshops, and team members are busy analysing photographs and building chassis. We thank the many people who have donated components, but I still need lots of half-watt transistors of 1940s or 1950s vintage.

By the time you are reading this there will be a little under two years to go when we want to re-run the world's first stored program computer on its 50th anniversary. A good start, but still a lot to do!

Systems Analysis for Business Applications

Frank Land

This article describes the author's experiences developing applications for Leo computer systems from 1952 to 1967.

Looking back at the early Lyons' applications developed for Leo I and Leo II, the characteristic which strikes one most forcefully is the very high level of ambition.

Current orthodoxy suggests that the pioneers took existing procedures and more or less automated them as they stood. But this was not what happened at Lyons. Each application involved what in current terminology might be described as business process re-engineering. I will use three of the early jobs as examples. The first, and perhaps most radical, was the Tea Shop Ordering job.

Analysis carried out by the Lyons Systems Research Office had shown that teashop manageresses tended to order to a pattern dictated by season and by day of the week. By and large, today's order was very similar to the order place on the same day the previous week. From this analysis stemmed the notion of the standard order. This idea led in turn to the concept of doing everything by exception. The concept of exception reporting had been used many times before, but not as part of the way a system was operated. The daily teashop order to the central kitchens comprised the standard menu plus or minus exceptions requested by each teashop.

Within this framework, the shop manageress had total freedom to order what she wanted to, although as with all systems running on the Leo there was always a possibility of an override, because of later information or because management wanted to do something special.

The second major novelty introduced by the system was the notification of the order data in as close to real time as it was possible to get with the technology then available. Each tea shop was telephoned at a certain time of day, when the teashop manageress would read out the variations from the standard order to an order-taker equipped with a card punch. Once the variations from the standard order for all teashops had been punched, they were entered into the computer for processing. Not all the tea shops had telephones: with those that did not, someone had to go to a telephone booth at a specified time, hope that there was nobody else in the way, and wait for a telephone call. Amazingly, the system worked!

Another innovation was the way the systems were integrated. We tried to use the data from the tea shop orders — a combination of standard orders and variations — to do calculations for the kitchens, for assembly of goods for each teashop in the order in which they were to be delivered, the allocation of appropriate packing materials, the assembly of pallets, and the provision of transport.

The system was designed to produce a wide range of management statistics intended to provide the local and senior management with information on each teashop's performance. However, the tea shop management had been trained to be good at man management, but were, with some exceptions, not skilled in business management. As a result, some carefully thought out management outputs had far less impact on the business than had been anticipated.

The second job I want to talk about is the reserve stores stock allocation job. At that time (all these jobs were developed in the early 1950s, from 1953 to 1956), rationing was still in force. Lyons had large stocks of materials in reserve stores for use in bakery, kitchen, confectionery and ice cream products. For example, to overcome the shortage of sugar, sweetened fat was used as a substitute. Large quantities were imported and held in the reserve stores.

This was a very ambitious application for its time. Again, we recognised production scheduling was a complex task, and that short term variations were inevitable. The system recognised that a simple automation of existing procedures was not possible. The system had to permit interventions, readjustments and reallocations. The system was designed to place orders on stores, taking into account interventions from management, and to arrange for the availability of transport for delivery of material to the factories.

The reserve store job was killed when rationing stopped and the reserve stores were no longer required. It had lost its *raison d'etre*. It was a nice job: from a systems point of view it was well conceived, but from a business point of view it had ceased to make sense.

The third job I want to talk about is the first for which I took major responsibility. It ran for 25 or 30 years. This was the tea blending job. It was not called a decision support system, but that is what it was. It gave management control over tea blending in two respects: the quality of the particular blend and the price. We had to reconcile the cost of the wide range of teas purchased at the Mincing Lane auctions or acquired on long term contracts from India, Ceylon and Rhodesia (as they then were) with the needs of the tea blenders who had to achieve a standard quality of product sold in the shops. In designing the application we worked very closely with tea tasters as well as the tea buyers and management accountants responsible for keeping the business profitable.

As part of the job, the movements of teas to and from bonded warehouses and the company's tea factories had to be planned and accounted for. Once again this was a very ambitious job, especially as it had to be carried out on a daily basis. We systems analysts and consultants considered what the job had to do and how we could possibly do it, given the limited amount of resources we had available. At the same time we tried to maximise what we could get out of any data once it had been prepared on punched cards or paper tape for the computer. Each day involved the discovery of new ways of doing things. In the excitement of finding a better way of doing something, our meal times and tea breaks were always full of discussions about what we had been doing.

We had no reference manual to tell us how to analyse applications, and there were no training courses in systems analysis available. Our system of learning resembled the notions of apprenticeship — a novice working closely with a more experienced mentor. We followed the way things had been done in the Systems Research Department, established before the second world war and later managed by David Caminer.

The main precepts of our analysis procedures derived from a long history of innovative system thinking, many of which had been developed by John Simmons. Some of the more important aspects of the practices developed in the Systems Research Department were:

- The need for close interworking between users and the analysts. Simmons very wisely said that there should be a symbiotic relationship involving a process of mutual learning.
- The recognition that ideas can come from anywhere, including one's peers, played an important role in developing successful systems. This is where our lunchtime discussions were so enormously helpful.
- That nothing should be done unless it could be shown to add value in some identifiable form. Each potential application had to pass the test of where and how it would be useful to the organisation.

- At the same time we consciously minimised risk, very often by carrying out experiments — doing what today would be called prototyping. An example was our introduction of document reading, which was to play a very important role in Lyons' applications.

The system we planned to introduce first involved order forms. The salesman had to make a mark on a piece of paper to indicate quantity — that is, instead of writing down the numerical quantity ordered, he made marks on the form in different columns to indicate the quantity required. The computer was to interpret those marks. How could we ensure a usable, reliable system?

We went into a long period of experimentation with different coloured forms and with different ways of marking. We went out with salesmen to see if they could actually mark these things reliably under working conditions. The experiments had to be carefully designed to achieve the outcomes we were looking for.

- Another important practice was keeping records of what was taking place. The management was absolutely insistent that we kept careful record of any decision we took, of what we were actually doing as well as the documentation of flow charts. They wanted a log of how decisions were arrived at.
- We had to understand the application from first principles. I think we were well aware that we could invent the way things should be done to satisfy our analysis of what the outcomes ought to be. But we were also aware that it is the details in practice which count. In the real world all sorts of thing happen which first principles don't really take account of.
- It was safer to start with the assumption that users were intelligent and did things rationally, than to assume the opposite.
- Our mentors stressed the importance of providing reconciliations based on standard accounting and auditing practice. Again and again it was drummed into us the importance of ensuring and visibly showing that the job has been correctly done, and that all the elements in the job balance together.

Quite recently, in 1993, a very large multinational company had to abandon a major MIS system, costing many millions of pounds, because it provided no reconciliation of income and expenditure flows between different business units of the company. They could not make those things balance. This could not have happened with the kind of thinking which we put into the Leo applications.

As we gained experience in using the Leo computers we added to our understanding of analysis. Application analysis attempted to recognise:

- What could be squeezed out of the underlying transaction data which would add value to the business. Hence most applications we developed for J Lyons had elements of what today would be called transaction processing, management information systems, decision support systems and even executive information systems.

- Which attributes of a system were likely to be permanent and which were likely to change through time. The latter were dealt with by designing special systems-change data forms rather than by implementing expensive and error-generating program changes. We were not always successful in second guessing where changes might be required, but we did manage to devise flexible systems.

- The importance of building restart procedures into any computer-based systems. As a result we achieved a very high reliability rating in the delivery of computer results, despite working with computers very much less reliable than those available today.

- The importance of providing back-up facilities to protect the user against unforeseen accidents or catastrophic failures. Equally important was the need to keep back-up facilities up to date and to test them under realistic simulated failure conditions.

- In most systems a relatively small number of all possible types of transactions account for a significant part of the total volume or value of transactions. The so called Pareto Principle suggests that often 20% of possible transaction types account for 80% of the total value. A well known design principle states that it is possible to save very substantial design and implementation costs by focusing the design of the computer system on the most active 20% of transaction types. However, we learned that the designer has to balance the reduction in design and implementation costs against the increased costs of operating the system, stemming from the need to synchronise and coordinate the automated and non-automated parts of the system.

> In practice coping with the costs of synchronisation and coordination and correcting the consequential errors can wipe out many of the expected benefits of the application.
>
> It is interesting to note that David Caminer wrote an article in the very first issue of the *Computer Journal* making the case for going for the comprehensive application, and suggesting that many failures — already the subject of analysis in the *Computer Journal* in 1964 — were due to the problems of incomplete systems.

We incorporated the methods we were applying into our training courses, so that by the 1960s systems analysis was an important component of courses run by Leo. We gradually replaced the apprenticeship system, and I think it was a loss. Training courses are fine but the loss of the apprenticeship system made it more difficult for future generations to get the kind of understanding we took for granted.

At the same time we had to guard against the problem of 'analyst arrogance'. The feeling that we and only we had the key to the Holy Grail of successful computer application development. We looked at the mess other people were making, and noted that many people were not using the standards and notions of quality we had developed.

In the end we were facing totally different situations. Whereas in the 1950s we worked with customers to produce a plan on how to use the computer productively in the user organisation, in the 1960s the user company sent us standard specification documents, or standard benchmarks, and all we had to do was to provide them with a costing of a Leo configuration to do the specified portfolio of jobs. The potential customer was not interested in our view of how the jobs were to be done. In any case our interface with the customer was no longer a potential user but the client company's own computer department.

And so gradually over the 1960s, the way things had to be done altered, and in particular instead of seeing our role as primarily systems analysis, they changed to the more conventional role of computer salesman. In some way we felt the new role deprived us (and our clients) of the benefits of our hard won skills. On the other hand we knew we could not resurrect the world of the pioneers.

Editor's note: this is an edited version of the talk given by the author to the Society during the Leo seminar at the Science Museum on 18 May 1995. Frank Land is Visiting Professor of Information Management at the London School of Economics.

Systems Design — Then and Now

John Aris

This article describes the background thinking behind the systems design procedures employed by the Leo Computers team. The author shows how technological advances together with increasing experience have led to substantial differences between the design of office systems in the late fifties and early sixties and the same activity today.

When I joined Leo Computers in 1958, the reigning machine was the Leo 2-1. It was a small machine in everything except physical size, and the peripherals were limited to punched cards (in and out), paper tape (in) and a printer (out). It had no magnetic tape, indeed no backing store of any kind.

At the time there was an unwritten but fairly well understood set of principles which guided systems designers. These were passed on in a semi-instinctive apprenticeship fashion, and I have tried to abstract the design principles from how we worked.

Systems Design Principles 1958

- design radically
- check everything
- document design is crucial
- specifications must be detailed, agreed and frozen
- computers are expensive, programmers cheap
- configure cautiously

Figure 1.

One thing unique to the Leo operation was that we did not come from a background in punched cards or accounting machines, and therefore had no preconceived ideas based on these of how to design systems. So we tried to do a thorough job on the basis of the underlying requirements, with regard to what customer practice was, as well as what the ideal solution should be.

Checking everything was a long-standing Lyons tenet. That was essential with valve machines, which did break down rather often. It was also an article of faith that program errors were the fault of the programmer, so we went to considerable lengths with desk checking and program testing to try and make sure that bugs didn't get through. They still did, but in limited numbers.

As far as documents were concerned, the principle was that they should be useful. It didn't matter whether they were congested payslips with nothing but numbers on them, or turn-round forms for tea shop manageresses, or forms for salesmen to mark up, they should not lead users into unnecessary errors. We recognised that this was crucially important, and went to a lot of trouble to avoid it.

The spiritual heirs to that approach nowadays are the man-machine interface specialists. I think this issue is still not given its true importance by the world in general: it is just as important today as it was in the fifties.

The specification document was enormously important. The amount of detail needed to specify a computer system was only starting to be understood at the time. The "broad view", as typically seen by management, was of course not enough. We took the view that the specification must be both agreed and frozen before we started work. This is an interesting issue which provokes passionate views pro and con to this very day, and I will return to it later.

At the time, machines were expensive and people were cheap, so if you were able to use plenty of time from very bright people and, for example, get them excited about about the technicalities of shortening the inner loops, thus reducing the workload on the machine, that was an excellent bargain. That is a principle which we have completely abandoned today. Machines are now cheap and people very expensive.

We felt very conscientious and very confident in telling a customer just how much machine he was going to want and how much it was going to cost him. We looked pityingly at the poor customers who were duped by our competitors and bought a machine which turned out to be unable to do the job and then, instead of turning on their supplier and throwing him out, merely bought lots more of his kit.

We supposed that virtue would be rewarded, and that our intelligent customers would say "They're honest and they know what they are doing. Despite the fact that it costs another £100, we'll go to them". Somehow that didn't often happen. One doesn't always give the job to the plumber who comes round to do an estimate, looks at your boiler and sucks his teeth as if to say, "This is going to be expensive". We were rather too often in that position, but we did genuinely try to configure machines that really did the job. I would say we had a better than 50% success rate.

Behind these design principles there were three underlying axioms, which may seem strange now but which we did not question at the time. The first was that the supplier knew best. Most of our relationships with customers were like that in the early days, and I think it was the right attitude, certainly until the early sixties. Up till then it was rare to encounter a customer who had the experience and background knowledge.

What the computer industry as a whole, and not least Leo Computers and its successors, were slow to realise is that the situation changed. By the mid-seventies the users knew most about the jobs and how to do them, and the suppliers' knowledge had become more restricted to the technological domain. I think that is an important difference between 1958 and today.

The second axiom was that every problem is unique. Every customer had his own needs. Their systems were obviously unique in detail, and one of the things we didn't spot at Leo Computers until quite late was the possibility of standard applications package software.

That perhaps was due not so much to blindness on our part as to the spirit of the age. I don't know when the standard payroll program actually came on to the market — I would guess it was the mid-sixties. Maybe, with our experience, we should have thought of it and done something about it before anyone else, but it was the uniqueness of systems which struck us rather than their similarities.

The third axiom which went unquestioned was the issue of business sponsorship. Contrary to our belief, top management (with rare exceptions) really wasn't very interested in computers. They recognised that computing sounded interesting and was worth a bit of research and investment, but they thought of it as an indulgence rather than something that was going to be part of the fabric of the business. So they delegated their computer usage to bright middle managers.

At the time that was not inappropriate, because the justification for investing in a computer was typically for projects which offered demonstrable savings in staff or stock levels. Getting a middle manager to take charge of it with clear cut objectives laid down was reasonable. There wasn't at that time, again with rare exceptions, the idea that the use of computing systems might actually turn the business upside down and inside out and that that might be beneficial.

At Leo we weren't conscious of this overall picture: we came from Lyons, and at Lyons the business sponsorship was in fact there. We tended to assume it was there in other places where it wasn't. There was one of the early installations which came badly unstuck because, as we discovered the hard way, our customer — the person with whom we dealt, the person who actually signed the order and the cheque — turned out not to have the clout and the knowledge of the requirements of the organisation which were essential to get the application working. I think we were quite lucky that I only recall that happening once. The world again in the 1990s is very different: we are all conscious how essential sponsorship is.

Detailed design

Moving to a level of detail below the system design principles, there was in 1958 a series of rules of thumb for detailed design. Detailed design, I think, we were particularly good at. It required a lot of ingenuity and creativity, and it was great fun to do.

At that time, with no mag tape or backing store, the crux was file design. The requirement was to make sure that the files were optimally set up. They would be on punched cards in binary, so designers were constrained by the shape and size of a punched card, and by the need to express things in as short a binary field as you could get away with.

The fundamental of good system design was to define your files and their contents and their relationship to one another: then you could minimise the number of computer runs and make them most effective. The files had to be as short as possible because reading at a rate of 200 cards a minute took a long time, and output was even slower at 100 cards a minute.

So there was an absolute requirement on the system designer to minimise the number of times he had to pass through files, particularly long files. You also had to make those files which you passed through most often as short as you could.

The other main constraint was that Leo 2-1 had a ridiculously small memory: the capacity was 2000 short words (of 19 bits) and into that you had to put all your working areas and program. Nowadays that seems appallingly constrained but at the time one lived with it and designed around it.

Sorting out the file and program structures in the light of those constraints was the first and biggest issue. Then we had to worry about the input medium. Paper tape allowed you to get in the maximum amount of information in the shortest amount of input file passing time, and you could use variable length data. What you couldn't do with paper tape was sort it, and sorting on the computer itself was impossible without mag tape or backing store. So any sorting that had to be done required a card sorter, with the data on ordinary decimal punched cards.

System design revolutions

In the period immediately after 1958 systems design was affected by two and a half hardware revolutions and two software revolutions.

The hardware revolutions were the arrivals of mag tape, of multiprogramming and of random access devices.

Mag tape relieved many of the file handling problems I have discussed, but nonetheless we continued keeping mag tape files as short and as tightly packed as possible. The technology changed all the parameters, but the basic principles remained.

Multiprogramming allowed us to run programs in parallel. In theory you could run any program in parallel with any other, but in practice you had to divide your machine notionally into streams. Typically you had an input stream which involved the input devices, a mag tape deck and a chunk of store in which to hold the program. There was the printing stream which required the converse. The rest of the machine's resources were used for big record updating runs and for sorting. That sort of job layout I think can be used to this day.

Random access was interesting because it looked like the answer to a maiden's prayer; it got round the requirement for file passing altogether. If you could access just the records you wanted to directly, it saved all the overheads, didn't it? No it didn't, because it took 20 years to solve the problem of knowing where the record you wanted was, and getting at it in an efficient manner. Virtually all the early systems based on random access devices treated the random access files as sequential. The virtue turned out to be that you didn't have to keep changing mag tapes.

In terms of Leo there were two separate software revolutions a year or so apart.

The first was the arrival on Leo 3 of an operating system: this identified procedures that were common to all applications, but which up till then had been elaborately written individually (for handling input and output devices, providing information to the operator, restarting after a breakdown and so on), standardised them and packaged them all into one piece of software.

At the time senior programmers became very niggly about this because it took up 4K words — £15,000 down the drain! What is more the machine ran a bit slower and the software took away quite a lot of interesting things that we had enjoyed programming in the past. I suspect that wounded professional pride came into our reaction as well.

One learnt rather quickly that actually an operating system is infinitely better for all concerned — operators, users and programmers. But then all of a sudden there was another dose of the same nasty medicine for the programmer.

That was a high level language, CLEO, similar to Cobol and brought in at a time when Cobol was beginning to make significant headway in the States but was not that well known here. It was a novelty, and as with the operating system it took up more space, and space was what we'd been trained to economise to the utmost. Again, too, it took up running time and it offended professional pride.

What really happened in those two revolutions was that programming had started down the path of de-skilling. Till 1961 at least programming had been immensely intellectually challenging and enjoyable, and as the job was combined with systems analysis, that made it even more so. Now systems analysis was still there but programming was becoming dull. Perhaps that has endured to this day.

Systems design today

With these hardware and software advances, system design in 1965 had changed a lot since 1958, but nevertheless the same basic principles still held. So now let's look at the principles for system design in 1995 (see Figure 2).

This time I can only give you a list of issues, not design principles, as the enormous diversity of what is done on computers has by now revealed that there is no one best way of doing things. There is a spectrum of good practice, and depending on one's circumstances one picks a particular spot on that spectrum. That choice will depend largely on how much business sponsorship there is — there will be a best practice somewhere between simplicity and ambition — and how radical a redesign one is attempting.

Systems Design Issues 1995

- business sponsorship
- project selection and justification
- simplicity versus ambition
- radical redesign?
- hard versus soft specifications
- management information
- databases
- sizing
- outsourcing

Figure 2.

Business sponsorship was not a question we asked about in the fifties. Systems design nowadays, in contrast, does depend very much on the degree of commitment at senior managerial levels. The design issues include the basic problem of how to get it!

In 1960 projects to some extent selected themselves. If you were willing to consider a computer it was pretty obvious you were going to get some benefit from doing payrolls, or distribution systems or invoicing systems. In most cases you would save a lot of labour; the justification could be expressed in cash terms. More recently projects have been less demonstrably justifiable in most cases. That is now an issue with which we struggle considerably.

In today's world there are diverging pulls towards simplicity of systems and ambition of systems. Some of us have burnt fingers proving that trying to implement elaborate and ambitious systems is painful. On the other hand the sheer availability not only of cheap hardware but also of programming tools tempts us into being ever more ambitious. That's an example of there being no right answer.

The same applies to the issue of radical redesign. The Hammers and Champys of this world would have us believe that if we are not radical we are nothing. Leo was advocating radical thinking back in the 1950s, but what Leo meant by radical redesign was not in fact what Hammer and Co mean by it: we did not aim to restructure our clients from the roots up.

The trouble is that Hammer and Co have become fashionable and nowadays everyone I talk to who is implementing a new system describes it as being business process redesign — whether it is or it isn't. Radicality really has to be thought about hard and judged in the light of particular circumstances. There are circumstances in which it is right to be exceedingly radical, though more frequently it is not. Of course top level business sponsorship is essential for radical design.

As to the issue of whether you can agree and freeze a detailed spec, in the state of technology in the fifties we had to. It was not on to do anything like modern prototyping, or even to proceed by trial and error.

Now, I think, one can classify systems. At one extreme there are systems which have to have a very hard spec, such as those that control nuclear plants or aircraft in flight. They had better be very exactly specified and very accurately implemented or there is literally going to be blood all over the place.

However, that type of application is fairly untypical. In commercial systems, Leo's home ground, most systems are towards the other end of the spectrum, where what you want to do becomes clear as you do it. Elements of trial and error and prototyping are very desirable. The big mistake is to use a soft spec when a hard one is needed or *vice versa*.

One unfortunate fact of computing life we've always recognised is that one of the most valuable things a computing system should be able to offer us is management information, but managers seem strangely ungrateful. Computer-produced information is unfamiliar; it doesn't help; it doesn't respond to the questions they are used to asking and they are (quite legitimately) busy and so don't want to rethink what questions they ought to be asking. We haven't cracked that one yet.

What about databases? They became a great fashion in the late sixties and through the seventies. It was conventional wisdom to stick all the available information into a database and then goodness knows what you can't do with it. For that matter goodness knows what you can do with it — even supposing you have succeeded in capturing all the data in the first place.

I think that's a fashion which has come and gone. Of course you need to think in terms of database technology, and to consider what you need to hold in the way of data in a computer system over and above the obvious. But the idea of being wholly comprehensive about it is no more sensible now than it was in the fifties.

Sizing, the question of configuring correctly, was important in 1958. Hardware was very expensive: a module of memory on Leo 3 was priced at more than £100,000 in today's money. So sizing was something we took very seriously indeed. With rare and difficult exceptions (some real time systems) that's gone away. It was fun in its way; it was an intellectual challenge but I think the human race is probably better without it.

Outsourcing is in a sense what Leo was doing. One of my 1958 underlying axioms was that the supplier knew best. And what the typical Leo customer was doing was buying a solution — not just a chunk of expensive hardware but the knowhow to put it to good use. That's not too bad a description of outsourcing as now seen.

The interesting thing is that that sort of outsourcing disappeared completely from the scene in the late sixties and has only come back in the nineties. If outsourcing was a good idea 25 years ago and it is a good idea now, why wasn't it a good idea in between? Did we at Leo Computers actually miss a market opportunity: was there a niche that we should have filled?

Editor's note: this is an edited version of the talk given by the author to the Society during the Leo seminar at the Science Museum on 18 May 1995. A fuller treatment of the subject can be found in the Book "User-Driven Innovation", published by McGraw-Hill, ISBN 0 07 709236 8, price £35.00.

Linear Programming and all that

Robin Judd

I first "got involved in computers" in the spring of 1959. My employer, J Bibby & Sons, had two successive Zebras, which I have designated in this story as Zebra one and Zebra two. (Their actual production numbers are unknown to me, but they were not the first two produced by many months.)

I lived and breathed the Zebra for nearly three years. The lessons I learned then have stood me in good stead for over 35 years and I guess they will stay with me until I stop programming computers and take up some less testing way of living, like retiring.

J Bibby & Sons was an old-established manufacturer of animal foodstuffs, cooking fats and soap. There were 17 men surnamed Bibby on the Board of Directors and one who was not, called M Morrice, who was the Company Secretary.

I was employed as an assistant statistician in the Development Department, although my formal exposure to statistics was a short course for scientists at university as part of my (very ordinary) degree in natural sciences and a short evening course at the Ashley Down Technical College in Bristol (now the University of Bath!).

I had heard about computers at school — I read "Discovery" and "Endeavour" — and had seen the special purpose computer that played noughts and crosses at the Festival of Britain in 1951 — I think it was made by Ferranti. I was not impressed. Computers were mentioned once or twice at university, but we students never saw one.

After the man who interviewed me, David Spurrell, had left Bibbys, I found his notes of my interview. He was more impressed with my farming background than my knowledge of statistics. This was appropriate enough for a job in which the nutrition of pigs loomed very large indeed. David had inherited the Company Statistician job at Bibbys from Andrew Muir, who had persuaded the company to buy the Stantec Zebra from STC for £30,000. To put that in context, my starting salary at Bibbys was £850, so a Zebra would be about half a million pounds at current prices.

The justification for the purchase of the Zebra was Linear Programming (LP), an iterative technique to enable you to calculate the least cost mixture of a number of components, each with different prices, each component containing different proportions of sub-components, subject to constraints. The classical example given in the original papers was to make a least cost mixture of nuts using several different sorts of nuts, with different prices, subject to minimum and maximum constraints on the amount of protein, fibre and starch.

Before the Zebra was purchased Andrew Muir had carried out a long series of feasibility studies. To simulate the LP algorithm running on the Zebra, two comptometer operators were employed. These delightful "comp-girls" could enter data, add and subtract using touch only, on comptometers. These were fast desktop electric calculators, virtually silent and often used behind the scenes in financial organisations.

Comp-girls knew nothing of mathematics and had to be provided with scripts just like computer programs, which they obeyed faithfully and with impressive speed, copying down results from the single register onto worksheets at the end of each iteration. The profession of comptometer operator and the comptometer itself have both long since passed into history.

Andrew Muir had written the LP program for the Zebra in Normal Code. This used the simplex method and the product form of the inverse. The basis for all this was a research report and several papers by GB Dantzig. LP was used routinely to calculate least cost mixes for a hundred or so different products ranging from layers' pellets (for chickens, to persuade them to lay an egg a day) to elephant nuts (for maintenance of the main attraction at a travelling circus).

Each case of animal feed took several minutes to solve. Being an iterative technique it was not possible to forecast how long it would take — the better the approximation you started with the quicker the program converged. The solutions were passed to the Nutritionist, a veterinary surgeon, who in turn passed them (or not) to the shift production foremen, who were responsible for the machine settings.

There was a problem here; the whole purpose of the operation was to deskill the nutritionist post which had always had considerable power, and salary, within Bibbys. However closely the specifications were set there were many, many occasions when the LP formulation was rejected on non-quantifiable grounds — colour, texture, feel and so on. I believe that, by the time I arrived, it had been decided not to use LP as a production process, and that was why Andrew Muir had left.

Nonetheless the Zebra ran overnight on LP on the eighth floor of the office building in Liverpool in 1959, 100 yards from the Head Office and directly connected to the factory itself — a noisy, dusty, smelly place where khaki-clad operatives laboured in conditions that had not changed significantly for 50 years. That is not to say that Bibbys was a backward-looking company. It had one of the first non-contributory pension schemes for all employees which contributed not a little to the tradition of long service in the company at all levels. To work at Bibbys in Liverpool had somewhat of the stability of a job in the civil service in a community where the depression of the 1930s seemed only yesterday.

The LP program was not without its problems. In particular the use of fixed point arithmetic on 32-bit numbers could cause over- or under-flow in certain cases. Since the LP algorithm used was independent of the scale of the intermediate results such problems could be resolved when signalled by rescaling, multiplying or dividing the offending row of the matrix by a factor of 100, and remembering to rescale the result. This was not automatic. If the LP program detected a need for a rescale it would stop. If an operator was present he would load the rescale paper tape program and try again!

As a new recruit who had read about floating point, I asked why this was not used; it seemed to me to be obvious. I learned about the restricted speed of the machine — 312 microseconds for a basic operation, but, with no hardware to help, 20-40 milliseconds for the floating point operations. The eight hour shift would have stretched to nearly a month!

The other main application was the forecasting of sales of the various animal feeds. These were seasonal and subject to variations due to price fluctuations in the products such as milk, bacon and eggs (although in the late fifties these were smoothed out by massive government intervention through product marketing boards). Sales forecasting used exponential smoothing, a method that minimised the amount of historical data that had to be kept.

I am not sure who wrote our exponential smoothing program, which was called EXPS and was based on an algorithm due to RG Brown of Arthur D Little. I suspect that the version used when I was at Bibbys was a joint production by Andrew Muir and David Spurrell. Again it was written in Normal Code for speed and space considerations. For each run the program and data were read in on 5-channel paper tape.

As I said earlier, I had never seen a real computer before, except in photographs. Again I was less than impressed. It seemed to me that it had little relevance to the work I was supposed to do, the design and analysis of animal nutrition experiments. I was soon to be re-evaluating my position.

In fact the design of the experiments was the least of my worries. How to layout the experiments was well understood — it was heavily constrained by the animal houses, long since built, and the carefully bred strains of animals, genetically similar to each other and hence capable of displaying small differences in weight, milk yield or backfat when fed slightly different rations.

It was the analysis that took the time. This was done using a technique due to RA Fisher called "Analysis of Variance", designed to pick up differences in response to a series of treatments in the presence of background variability, such as is always found in measuring any parameter of a living population.

Analysis of variance on a typical experiment could take days. The algorithm was not self-checking and although a desk calculator was used the entering of the data and its manipulation were severe tests of my accuracy. A full analysis could easily take a week as I found it impossible to calculate for more than about two hours at a stretch without introducing errors.

My first few months at Bibbys were a nightmare of calculation. I worked in a room with four engineers driven mad by the noise of my electromechanical desk calculator. On several occasions I inadvertently divided by zero, causing the mechanism to enter a noisy loop which could only be broken by disconnecting the mains, which jammed the machine. The manufacturer's maintenance engineer had to be called to disassemble the cog wheels and perform a hasty rebuild. Meanwhile the computer sat upstairs for long periods waiting for the price information to enable it to run the LP program or the final week's sales figures to allow the EXPS program to run.

I discussed with the Company Statistician whether it would be possible to write a program for analysis of variance. He was discouraging. I had still not cleared up the backlog of experiments to be analysed that he and Andrew Muir had allowed to build up during the writing of the LP and EXPS programs.

They had regarded these programs as essential to the future of the company: a 1% or 2% saving in raw material prices would have an immediate effect on the profitability of the company. It was difficult to see how detecting the significance of a small increase in the effect of layers' mash could be translated into profits.

Farmers were notoriously reluctant to change the feed they used for their animals and certainly would be unlikely to pay more for something with only a "scientifically" proven effect. The major increases in yield in farming were a joint effect of genetic programs and better formulations. Farmers could see the effects of the genetics and could understand well the effect of using a better breed of boar or bull, but the feeds all looked the same, just hessian bags full of something compounded by the miller.

During a rest from adding and squaring on the Muldiva I went up the eighth floor and talked to the third member of the computing section, Tom Corless. Tom was an acknowledged character known throughout the factory. He looked and sounded like a Liverpudlian burglar and always wore black boots with aggressive external steel toe-caps.

I asked Tom if the computer could add and square, the basic operations for the analysis of variance. He reassured me. I asked him if he could teach me how to program — Tom was clearly not overjoyed with the idea of spending time with this university graduate who did not, he quickly discovered, have anything like his acquaintance with Chrystal's Algebra (1865), his preferred lunchtime reading.

Both the Nutritionist and the Commodity Buyers were putting up a spirited resistance to any further incursion of the Zebra into their empires. The Bibby Zebra had always had a bad record for reliability. Often the night shift, already subject to stalling for rescaling, was aborted due to hardware error.

Suddenly a computer consultant appeared, called RH Williams. He had been a computer saleman for Univac in the United States, returned to his native Wales and set up a one-man computer consulting company. As a salesman he was no doubt successful but his knowledge of computers was restricted to the Univac and he didn't feel it necessary to have any technical expertise at all. As he explained it to me, he was involved in strategy, not tactics. Rumour had it that he had met one of the director Bibbys in a London club and was invited in to resolve the computer problems we were suffering at the time.

Considerable pressure was brought to bear on STC behind the scenes and a replacement Zebra was supplied. It was hoped that better reliability would overcome the objections of the user groups.

RH Williams was in charge of "Acceptance Tests" for the Zebra two and quickly displayed his lack of technical knowledge, instructing Tom to get it to multiply two numbers together after having removed the software multiplication routines. I spent some time talking to him and was underwhelmed, to such an extent that I went to "my" Director, Ben Bibby, and complained bitterly.

With the replacement machine installed, Williams displayed some ingenuity in producing a new scheme for least-cost animal feeds which could be run in parallel with the LP program. This consisted of simply costing all previously made formulations using current daily prices and printing off the 10 cheapest. Since these existing recipes had all been previously accepted by the Nutritionist there could be no dispute with him and he could simply select the one to be made in the factory from the candidates suggested by the computer. But, of course, the formulations might not be the cheapest possible: the Nutritionist could still explore other possibilities and suggest another recipe that could be added to the history file.

The problem with this method was the volatility of the commodity prices and the appearance of new raw materials to use in animal feedstuffs. This naturally generated new recipes and the history file, again kept on 5-hole paper tape, grew and grew. Eventually it exceeded three feet in diameter and had so much inertia that it had to be spun by hand at the start of the program to avoid being torn as the first recipe was read. The size of the cartwheel of paper tape was so great that even the company directors could see that the whole thing had got out of hand.

For whatever reason, RH Williams' contract was terminated. He later came to public notice when he sued the clearing banks unsuccessfully for royalties on the OCR numbers printed on the bottom of cheques. David Spurrell resigned at around the same time, and I succeeded him as Company Statistician.

I had my own interest in Zebras one and two. Although I had not quite cleared the backlog of experimental analyses, I persuaded the Chief Engineer to let me go on a programming course at STC in Newport.

This was a personal disaster at first. There was no real attempt to explain the basic ideas behind programming: we were, instead, led through the coding of what would now be described as the operating system! This was definitely not what I wanted nor could I understand what was going on.

In the second week we were introduced to "Simple Code", the interpeter which caused the Zebra to execute something very close to Edsac code. This was more what I was after and I began to see how to do real work on the machine. What is more, the default arithmetic type in Simple Code was floating point! (The Analysis of Variance, being concerned very largely with squaring and adding, caused overflow very quickly in fixed point).

I spent many evenings at home struggling with programming and eventually got the hang of it. I started off with the simpler things and wrote programs for mean, standard deviation, linear regression and Pearson correlation coefficient.

Analysis of Variance was my target and eventually, after attending a conference and getting some clues from JC Gower of Rothamsted, I managed to get the algorithm straight and programmed and debugged it. This was my first major program, called Anova. I was immensely helped by discussions with Tom Corless, whose capacity for abstract thought while lying full length on the computer room floor never ceased to amaze me.

This business of lying full length was often resorted to. A print-out of the program had to be done on a telex machine which used rolls of unperforated paper. These curled up unless weighted down and for programs of any size, say eight feet of print-out, horizontal on the floor was the best way to read them.

Like telex rolls, 5-hole paper tape was not a friendly medium. Apart from the fact that you could easily cut yourself on the edges of the tape and that short lengths would undulate snake-like along the floor until absorbed by the fans revolving silently inside the operator's desk pedestal, a mistake in punching a tape was a disaster.

Did one punch the tape again? No, one did not. One cut out the offending frame(s) and spliced in corrected frames, using a jig, transparent tape and emery boards — the sort used by comp-girls to file their finger nails. The emery boards were used to rough up the sellotape to enable the friction rollers to get a grip as the reader took on the spliced join. The cylindrical knives that cut out the confetti from the tapes — in demand for weddings — sometimes, after prolonged use, would produce the dreaded "trap-door" fault, where a hole would be opened or closed randomly on successive reads, the chad remaining connected to the edge of the hole by a tenuous but strong fibre.

I was, with startling suddenness, put in charge of Zebra two, the comp-girls and Tom Corless, a transition I was quite unprepared for. What was worse was that the position of the full scale implementation of the results of the daily LP runs and the weekly EXPS run was now untenable. The combined forces of the Nutritionist and the Commodity Buyers, who had no intention of allowing their grip on the company to be loosened by a computer, had forced a stalemate. I did not understand either the LP or the EXPS programs, which Tom now ran on the Zebra — and I wasn't interested in them either. It seemed to me a lost cause, far too difficult to explain to the company directors.

I can't remember much of the year I spent running Zebra two. I wrote a program which worked out targets for salesmen from the Ministry of Agriculture's census returns of animals in the various counties of England and Wales. All this did was cause alarm and despondency in the sales force; it made no allowance for the density of the populations or the average size of herds. Salesmen visited farms, not the animals themselves. I comforted myself with the thought that at least I hadn't specified the problem, only programmed a solution dictated by the sales manager.

It was in this period that I became interested in nomograms and simple devices for analogue computation, circular and spiral sliderules. With the aid of Zebra two I wrote a program to design circular sliderules for special purposes and this was converted into a "give-away" circular slide-rule which worked out the cost of producing a dozen eggs given a few parameters, including the cost of a ton of Bibby's Layers' Pellets. This was a great success and several thousand were distributed. I have always regretted not having kept one!

I ran the Anova program regularly and extended it to handle covariance. My work on analysis of experiments now took 40 minutes on a Monday morning and I had the rest of the week to think of other things. I ran all the old, hand-calculated, analyses through to check out the program and found several discrepancies in old reports. In every case these proved to be mistakes in the human computations, usually slips in decimal point location.

By now I had discovered the literature of computing, such as it was, and had decided that there was a need for something easier than Normal Code and more flexible than Simple Code which would cope with commercial problems. (Simple Code, like the Edsac machine code on which it was modelled, had been designed to facilitate matrix computation.)

I found literature on Algol and went on a Zebra Algol one-day course. We were promised an Algol interpreter for the Zebra, which I believe was being written in the Netherlands. The instructor refused to answer, or even discuss my only question — how do you read and print one character? I had discovered the theology of computer languages! In Algol, of course, input-output is never mentioned; it is a bit like discussing republicanism at a Royal Garden Party.

It was a case of back to the library, where the Bibbys librarian helped me to obtain various research reports from America. This led me to the early work on Cobol — I think it must have been, at this stage, without a year suffix, but the report I have is dated 1958. This seemed to be what I wanted but I was daunted by its size.

After some struggling I decided to subset it and write a translator for Zebra. I christened my new language Intrinsic, which was Bibby's telex answerback. After some further reading I decided to write it in its own language and bootstrap it up after the model of Halstead's Neliac, which was a quick and dirty attempt to produce an Algol clone. Halstead worked at the Naval Electronics Laboratory in the USA and the "iac" stood, I think, for International Algebraic Compiler. (The original name for what became known as Algol was IAL — International Algebraic Language.)

It was later that summer that the Chief Engineer found out what I was doing and gently told me to stop. This was too much and I started applying for other jobs — after all I could now call myself a compiler writer and talk about dynamic own arrays. I was sorry to say goodbye to Zebra two — like a first love I still have fond memories.

What happened at Bibbys? Years later, I tried to recruit Tom Corless to work for me at Trent Polytechnic, Nottingham. He reminded me about the non-contributory pension — I think he had about seven years to go at Bibbys. And he told me that he had translated my Anova program for the IBM that succeeded the Zebra. I was very pleased: by then the algorithm was 20 years old. I asked him about Linear Programming. "They don't do that anymore", he said.

Editor's note: this is an edited version of a talk given by the author to the Society during the half day seminar on the Zebra computer at the Science Museum in London on 11 October 1995.

Letters to the Editor

Dear Sir,

You have published a number of comments recently, starting with Gordon Scarrott's article[1], bemoaning the length of time it took for ICL to turn Cafs into a commercial product, and implying that the sole reason for this was a lack of vision by the company's management and/or marketeers. If we are going to study the history of the industry in order to gain insights that help us in the future, then I think a rather deeper analysis of this story is called for.

Let's start by noting that the timescale is not at all unusual. Relational databases, portable operating systems, object programming languages and the Internet are all technologies invented around the late 1960s that did not turn into successful commercial products until the early 1980s — exactly the same timescale as Cafs. Many other inventions of that period never made it at all: database machines, in particular, were a very fashionable research topic in the 1970s but one that made almost zero impact on the industry.

The most unique thing about Cafs, oddly, is that it is unique. Most research ideas either get copied or they get abandoned. It is hard to find another example in the IT industry of an idea that has remained unique to a single company from its inception in the research labs through to profitable maturity 25 years later. We need to try and understand why this should be.

The reason ICL found it difficult to exploit the Cafs idea, and the reason other companies found it impossible (we know some tried), is that it crosses architectural boundaries. Cafs as originally envisaged requires an intimate collaboration between the disc hardware and the end-user query language, a collaboration which the intervening layers of disc controllers, operating systems and data management software are almost deliberately designed to prevent. ("The purpose of an operating system", we were taught at college, "is to prevent applications from exploiting the hardware".) This meant that productising Cafs, when we eventually did so in the early 1980s, required an extraordinary degree of cooperation and coordinated planning between different hardware and software development teams organisationally and geographically scattered around the company.

This level of integration runs completely against the trend, which was

[1]"From Torsional Delay Lines to DAP" by Gordon G Scarrott, in *Resurrection* issue 12.

already evident at the time, towards building systems from standard off-the-shelf components: discs from one supplier, operating systems from another, database software from a third, query and reporting tools from a fourth. In retrospect, it seems a miracle that we succeeded at all. To make the concept viable, radical changes were made to the architecture, first to use standard disc hardware and block formats, and more recently in the current search accelerator product to make the engine microprogrammable, so as to accommodate the data formats and query semantics of third party relational database products. These changes sacrificed some of the performance benefits to achieve greater flexibility — a trade-off that is very much in line with industry trends.

The other important factor to consider is ICL's commercial priorities at the relevant times. Throughout the 1970s the R&D agenda was dominated by New Range. By the late 1970s I was the chief designer on the Codasyl database software, IDMSX, and we were working hard to improve transaction throughput and system recovery to meet the needs of real specific customers. We would have been much more enthusiastic about Cafs if it had helped us significantly in achieving these aims. Only when VME had caught up with the competition in these basic areas did we have the luxury of looking for competitive advantage in applications like name and address tracing at the Inland Revenue.

The real lesson of the Cafs story is that architectural innovation is much more difficult to bring off than ideas which replace a single component of a system with something faster, cheaper or more user-friendly. In today's open systems world it is becoming even more difficult than it was in 1980. It does require management vision, but it also requires a freedom of manoeuvre which few managements are lucky enough to enjoy.

Incidentally, there are a couple of good articles on the history of Cafs (factual rather than analytic) by Hamish Carmichael and Andrew Hutt in the November 1985 issue of the ICL Technical Journal.

Best regards,
Michael H Kay
ICL
Bracknell
Berkshire
15 January 1996

Dear Editor,

Jeremy Walker's piece about the Deuce in your last issue was most interesting on the period from when he became involved in the mid 1950s, but largely imaginary regarding earlier events; it was fascinating to observe how rapidly comparatively recent events can turn from history into mythology. Readers interested in a more contemporary account of the early history of the Ace and Deuce family can find plenty of references in *Resurrection* back numbers; these include pieces by Donald Davies (issues 2 and 8), David Clayden (issue 2), Ted Newman (issues 4 and 9) and Fred Osborne (issue 10).

One of Jeremy's stories at the end of his article reminded me of arriving at NPL in September 1950 to find a Pilot Ace which would only ever operate with one or more chassis protruding clear of their close-packed neighbours on special "standerpround" chassis designed to facilitate fault location; when all chassis were returned into line, the machine simply stopped working! A year later, they put me in charge of maintenance (possibly forming the world's first specialist maintenance team?), and instantly regretted this since I banned any attempt to use the computer until all units would work in their proper places. For ever — or so I thought until reading Jeremy's story about Deuces which would operate only with their doors open and battery boxes plugged in all over. We live and learn!

Yours sincerely,
George Davis
Croydon
22 April 1996

Editorial fax number

Readers wishing to contact the Editor may do so by fax, on 0181-715 0484.

Forthcoming Events

13-14 July 1996, and fortnightly thereafter Guided tours and exhibition at Bletchley Park, price £3.00, or £2.00 for concessions

Exhibition of wartime code-breaking equipment and procedures, including the replica Colossus, plus 90 minute tours of the wartime buildings.

1 October 1996 North West Group meeting

The KDF9 Computer 50 years on, by Bob Beard

26 November 1996 North West Group meeting

The Use of Computers in the Fifties and Sixties

The North West Group meetings will be held in the Conference Room at the Museum of Science and Industry, Manchester, at 1730 hours. Refreshments are available from 1700.

London meetings for the autumn are still in the planning stage. Queries about London meetings should be addressed to Chris Hipwell on 0182572 2567 or George Davis on 0181 681 7784, and about Manchester meetings to William Gunn on 01663 764997.

Resurrection is the bulletin of the Computer Conservation Society and is distributed free to members. Additional copies are £3.00 each, or £10.00 for a subscription covering four issues.

Editor – Nicholas Enticknap   Typesetting – Nicholas Enticknap
Typesetting design – Adrian Johnstone   Cover design – Tony Sale
Transcripts of talks – Pat Woodroffe and Hilma Quinn
Printed by the British Computer Society

©Computer Conservation Society

Committee of the Society